School of Physics and Astronomy

Senior Honours Project
Computational Physics

Simulation of Networked Quantum Computing on Encrypted Data

Ieva Čepaitė
December 2017

Abstract

Due to the limited availability of quantum computing power in the near future, cryptographic security techniques must be developed for secure remote use of current and future quantum computing hardware. Prominent among these is Universal Blind Quantum Computation (UBQC) and its variations such as Quantum Fully Homomorphic Encryption (QFHE), which herald interactive and remote secure quantum computing power becoming available to parties that require little more than the ability to prepare and measure single qubits. Here I present a simulation of such a protocol, tested classically on the simulation platform LIQ$Ui\,|\rangle$ and then later adapted to and run on the recently released IBM 16-qubit quantum chip using their beta cloud service. It demonstrates the functionality of the protocol and explores the effects of noise on potential physical systems that would be used to implement it.

Declaration

I declare that this project and report is my own work.

Signature: Date: 30th November 2017

Supervisor: Dr. E. Kashefi

Contents

1 Introduction 2

2 Background 2
- 2.1 Quantum Computing . . . 2
  - 2.1.1 Measurement . . . 4
  - 2.1.2 Noise and Decoherence . . . 4
  - 2.1.3 Entanglement and Bell States . . . 4
- 2.2 Measurement Based Quantum Computation . . . 5
  - 2.2.1 The Measurement Pattern . . . 5
  - 2.2.2 Flow and Graphs . . . 6
- 2.3 Quantum Fully Homomorphic Encryption Scheme . . . 7
  - 2.3.1 Universal Blind Quantum Computation . . . 7
  - 2.3.2 The Simulated QFHE Scheme . . . 8
  - 2.3.3 Corrections . . . 9

3 Method 10
- 3.1 LIQUiji . . . 11
  - 3.1.1 Interactive Corrections . . . 11
  - 3.1.2 Deferred Corrections and Implementing the Protocol . . . 11
- 3.2 IBM Quantum Information Software Kit (QISKit) . . . 12
  - 3.2.1 Conditional Measurements . . . 12
  - 3.2.2 Qubit Couplings . . . 13
  - 3.2.3 Final Optimization and Classical Post-Quantum Processing . . . 14

4 Results and Discussion 15
- 4.1 LIQUiji Outputs . . . 15
- 4.2 IBM Simulator Outputs . . . 16
- 4.3 IBM 16-Qubit Chip Outputs . . . 17
- 4.4 Results Post-Optimization and Using Classical Processing of Measurements 19

5 Conclusions and Future Research 20

6 Acknowledgements 22

Appendices 24
- A Samples of Code . . . 24
  - A.1 QISKit Code for IBM . . . 24
  - A.2 LIQUiji Code . . . 25
- B Common Quantum Gates . . . 27

1 Introduction

Quantum computers and quantum algorithms promise to usher in a new era of information processing, solving problems that prove intractable for classical computers. Whether it is the simulation of physical quantum systems[1] or factoring large prime numbers in timescales previously thought to be impossible[2], it is clear that quantum processors can and will be applied universally once the technologies for it are fully developed.

The fragile nature of quantum systems- particularly those used in information processing- means that access to these technologies, such as those developed by IBM[3] is likely to be limited. This has led to an interest in distributed quantum computation. In order for individuals and institutions to be able to use these processors remotely, through a cloud-based service for example, there exists a need for security protocols that would allow the computations to be carried out privately, without servers gaining any access to the nature of the information being processed, up to a certain minimal, leaked amount (such as the size of the input or output).

A novel protocol[4] for performing remote and secure quantum computations was introduced by Broadbent, Fitzsimons and Kashe in 2008 called Universal Blind Quantum Computation (UBQC). It has since been shown to be secure and correct, both in a standalone setting and as a cryptographic primitive[5]. Variations on this protocol have been proposed since, such as the Quantum Fully Homomorphic Encryption (QFHE) scheme. UBQC and QFHE will be described in more depth in the next chapter of this report as well as the model of computation they are constructed from- Measurement Based Quantum Computing (MBQC).

The goal of this project was to understand and classically simulate a recently proposed QFHE scheme, developed by P. Walled, M. Hoban, A. Gheorghiu and E. Kashe . The simulation was rst implemented using Microsoft's LIQUi ji [6] simulation platform and then adapted to be run on the recently released IBM 16-qubit cloud quantum processor in order to verify its outputs and investigate the impact of noise on such protocols where current quantum computing technologies are concerned.

This report will aim to guide the reader through the necessary background on Quantum Computation, MBQC and the QFHE protocol before presenting the methods used for the simulations and the results that they produced.

2 Background

2.1 Quantum Computing

The analogy to a classical bit in quantum computing is the *qubit*. It is a two-state quantum-mechanical system whose state can be represented as a unit vector in two-dimensional complex Hilbert space. The orthonormal basis in that space is usually labelled with basis vectors $|0\rangle$ and $|1\rangle$. Any state of a qubit can then be represented as a linear superposition of the two bases:

$$|0\rangle + \quad |1\rangle$$

where and are complex coe cients with the property $| |^2 + | |^2 = 1$. A tool to geometrically represent the states of a qubit is the Bloch sphere as seen in Figure 1. For a

two-dimensional Hilbert space, complex projective lines in the sphere are subspaces that all together represent all pure quantum states of the computer.

Figure 1: The Bloch Sphere: a geometrical representation of a two-level quantum mechanical system. Lines on the sphere represent two-dimensional quantum states with the North and South poles usually correspond to the $|0\rangle$ and $|1\rangle$ basis states respectively. Image source: https://en.wikipedia.org/wiki/Bloch_sphere

The basis $\{|0\rangle, |1\rangle\}$ is known as the computational basis of the quantum computer and computations are performed by applying unitary operations to it and changing the qubit's state. This is easily visualized as rotating the line of a quantum state on the Bloch sphere at different angles to new points.

The n qubit state is a unit vector in the tensor product space $H_1 \otimes H_2 \otimes \cdots \otimes H_n$. The 2^n basis states of this space are n-fold tensor products of $|0\rangle$ and $|1\rangle$, where $|0\rangle \otimes \cdots \otimes |0\rangle$ is written as $|0\cdots0\rangle$. Thus, any state of the quantum computer $|\psi\rangle$ is a 2^n dimensional complex unit vector that can be expressed as:

$$|\psi\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle$$

A unitary operation on this quantum state is usually represented as a quantum gate in a circuit, as seen in Figure 2. The gates are applied from left to right in time-steps until the computation is complete. These unitaries are often denoted by matrices that are applied to the state vectors. The gates used in this project and their notation are found in Appendix B.

Every computation realizable by a quantum circuit can be implemented by using a small number of gates called a *universal* set. This means that for any integer $m \geq 1$, any n-qubit unitary operator can be expressed by a quantum circuit using only a finite number of gates from the universal set.

Figure 2: A generic quantum circuit, where the wires represent single qubits and $|q_i\rangle$ are initial qubit states while $|o_i\rangle$ are the final output states. The rectangles are quantum gates, which can span any number of wires and apply specific unitary operations to the qubits they are drawn on.

2.1.1 Measurement

While the state of a classical computer can be read at any point during a computation without interfering with the process, observing the state of a quantum computer will immediately collapse its superposition of basis states into a single eigenstate. The measurement of a quantum system is essentially the application of measurement operators M_i with i di erent outcomes on the state of the computer $j\ \ i$. The probability of the measurement outcome depends on the pre-measurement state of the system:

$$p(i) = h\ \ j M_i^y M_i j\ \ i$$

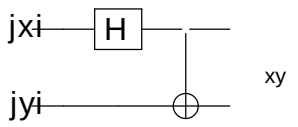
And the operators $f M_1; :::M_i g$ have to obey completeness. A special case of measurement operators are projection operators, where $M_i^y M_i j = i\ P_i$ and they are pairwise orthogonal: $P_i P_j = \ _{ij} P_i$. After a measurement occurs the state of the system becomes $p \frac{1}{p(i)} P_i j\ \ i$ and the change of the system from the state $j\ \ i$ to $P_i j\ \ i$ is not a unitary evolution but rather the collapse of the wavefunction. Thus measurement operations cannot be undone and lead to di culties in devising quantum algorithms.

2.1.2 Noise and Decoherence

Beyond the issues concerning measurement, a nal important thing to note is the concept of decoherence. This it the loss of quantum information over time due to the qubit's interaction with the environment. Quantum mechanical systems are incredibly volatile and most qubit technology relies on near-perfect isolation from any disturbances that might change the state of the system. However perfect isolation is impossible and a qubit's state will likely change or 'decohere' due to the time it is left in contact with its environment or the various operations that are performed on it during the computation.[7] There are quantum codes developed to deal with the problems caused by this[8] such as the infamous Shor's Code but, unfortunately, they require even more qubits to be introduced to the circuit, of which there is still a shortage in the world of technology.

2.1.3 Entanglement and Bell States

One of the most important quantum mechanical phenomena used in quantum computation is entanglement, best illustrated by the so-called Bell States. These are pairs of qubits that exhibit the Einstein-Podolsky-Rosen Paradox and are thus known as EPR pairs.[9][8] The concept can extend to more than two qubits and they are used extensively in the protocols discussed later in the report. The mechanism of creating a Bell pair of two qubits is shown in the diagram below:



$$j00i! \quad p\frac{1}{2}(j00i + j11i) = \ _{00}$$

$$|01\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = \Phi_{01}$$

$$|10\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = \Phi_{10}$$

$$|11\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = \Phi_{11}$$

The advantage of Bell States is not immediately apparent. They are states of composite systems that cannot be written as tensor product of the states of its components. Computationally, it can be seen that measuring the state of one qubit will immediately tell you the value of the other. In the state Φ_{00} for example, if the outcome of the measurement of the first qubit is 0, then the value of the second qubit can only be 0 too. This will be exploited in the models discussed in the following sections.

2.2 Measurement Based Quantum Computation

The content of this section is largely based on lectures by P. Wallden as found in [10] and the paper by V. Danos, E. Kashefi and P. Panangaden [11].

The MBQC model of quantum computation is realised by creating a large entangled state of qubits and then measuring them in different bases as described by angles on the Bloch sphere, in a particular order. This model of computation is equivalent tos the actions of quantum gates. The qubits are thus evolved unitarily one-by-one, where each measurement transmits information to the next qubit through the entanglement and then corrections are applied in the form of changing the measurement angles based on the flow of the computation. While the act of measuring a quantum system implies the loss of information, MBQC can still perform universal quantum computation. Furthermore, it can provide certain advantages in terms of architecture, fault-tolerance and cryptographic applications.

2.2.1 The Measurement Pattern

In order to use MBQC we first have to define the basis:

$$|+\ \rangle_i = \frac{1}{\sqrt{2}}(|0\rangle_i + e^{i\ }|1\rangle_i)\ ;\ |-\ \rangle_i = \frac{1}{\sqrt{2}}(|0\rangle_i - e^{i\ }|1\rangle_i)$$

We also define the measurement operation $M_i^{\ }$ which projects the qubit i onto one of the states $|\pm\ \rangle_i$ and the $\hat{\ }Z$ gate, which is a two-qubit gate used to entangle them. It is also known as the 'controlled-Z gate, since it applies Z a operation on a target qubit based on the (unmeasured) state of a 'control' qubit as defined in a circuit. Since its operation is symmetrical, it can be applied using either of the two qubits as the control for an equivalent result.

The classical outcome of the measurement on qubit i is denoted as $s_i \in Z_2$ and we define $s_i = 0$ if the measurement outcome after applying $M_i^{\ }$ was $|+\ \rangle_i$ and $s_i = 1$ for outcome $|-\ \rangle_i$. In an MBQC computation, all qubits start out in a $|+\rangle_i$ state and are then entangled into a large cluster state using $\hat{\ }Z$ operations, before being measured by applying the $M_i^{\ }$ operator with different values of $\ $ based on the desired computation.

These measurements then induce Pauli gate corrections in unmeasured qubits, where a correction can be an X or Z gate.

To see how these measurements perform the correct computation, we need to introduce the gate set $J(\ )$ (the set of generators $\{\hat{\ } Z; J(\ );\ \ 2\ [0; 2\ )\}$ is universal and can approximate any transformation in C^2, as discussed in [11]). A $J(\ )$ gate can then be turned into a one-way measurement pattern for any . If we de ne $\hat{\ } Z_{ij}$ as the $\hat{\ } Z$ gate being applied between qubits i and j and $X_j^{s_i}$ as a correction on qubit j based on the measurement outcome s_i, where $s_i = 0$ implies no correction while $s_i = 1$ would lead to applying an X gate to the indicated qubit to rotate it back to the required state. The pattern can now be written as:

$$J(\ ) = X_2^{s1} M_1\ \ \hat{\ } Z_{12}$$

The above consists of entangling two qubits, measuring the rst one at angle and then applying a correction on the second qubit based on the outcome of the rst. The second qubit is then the 'output' that carries the information of the applied gate onto the next layer of measurements.

2.2.2 Flow and Graphs

The ow is an important construct that de nes the order of measurements and corrections in MBQC. First we need to consider an open graph state (G, I, O) which consists of an undirected graph G as well as subsets of nodes I and O which are known as inputs and outputs respectively, with I^c and O^c being de ned as their complements. It is said to have ow if there exists a map $f : O^c\ !\ \ I^c$ and a partial order over qubits[12]:

- $x\ \ f(x)$: x and $f(x)$ are neighbours on the graph
- $x\ \ f(x)$: ($f(x)$ is to the future of x with respect to the partial order)
- For all $y\ \ f(x)$, we have $x\ \ y$: (any other neighbours of $f(x)$ are to the future of x)

Figure 3: An open graph state with ow. The boxed qubits are the inputs and white circles are the outputs. All the non-output qubits, black circles, will be measured during the run of the pattern. The ow function is represented as arrows and the partial order on the vertices are given by the 4 partition sets.[12]

The X and Z Pauli gates have properties:

$$M_i\ {}^i X = M_i\ \ {}^i$$

$$M_i^{\ i} Z = M_i^{\ i+}$$

So for qubit i the corrections can be applied by changing the measurement angle and accumulate as:

an X correction from $f^{-1}(i)$, the qubit whose flow is i

a Z correction from all qubits $j \neq i$ whose flow $f(j)$ is a neighbour to i

The actual measurement angles for qubits in the graph is then defined as:

$$\theta_i^0 = (-1)^{s_{f^{-1}(i)}} \theta_i + \left(\sum_{j : i \in N_G(f(j)); j \neq i} s_j \right)$$

Where $N_G(f(j))$ are all the neighbours of $f(j)$ in the graph.

These MBQC Graph states are often represented in diagrams where edges are entanglements and vertices are qubits as in Figure 3. To represent flow in MBQC computations, the graph states are often regular in shape and read from left to right, which is particularly important in large brickwork states as in Figure 4, which are constructed when MBQC is performed for UBQC, as discussed in the next section.

2.3 Quantum Fully Homomorphic Encryption Scheme

Now that the groundwork is set for the MBQC model, it is possible to introduce how it is applied in potential Quantum Encryption schemes and protocols. The most prominent of these is UBQC, which is discussed briefly below. However, many variations have been constructed since, such as the QFHE protocol that was simulated in this project.

2.3.1 Universal Blind Quantum Computation

UBQC was developed to allow a client (Alice) with no quantum computational power or memory to be able to delegate her computations to a quantum server (Bob) completely privately, without leaking knowledge about the input, output or even the computation itself. Alice's abilities would have to consist of being able to prepare single qubits with pre-rotated angles that Bob would then have no knowledge of.

Figure 4: A graph of the brickwork state as constructed for UBQC computations. [4]

Bob would then have to entangle these qubits in a pre-determined brickwork state, as shown in Figure 4 and discussed in [4]. This would inevitably reveal the upper bounds of the dimensions of the graph to Bob, but no information is available to them beyond that. The brickwork state can be created for any universal quantum gate by padding it with extra qubits so as to hide the shape of the graph.

Finally, Bob would begin measuring the qubits in layers according to the given ow and send the results of their measurements to Alice, who would then communicate the angles at which to measure the next layer of qubits. These would be computed based on the position of the qubit in the brickwork state and the padding she adds to each measurement by pre-rotating the qubits she sends at angles chosen from some set, as well as adding a single one-time rotations in a randomized fashion. Thus when she gets the results, she can apply corrections to the outputs to decode the results of her computation.

She can also verify that the computation being performed is correct by hiding 'dummy' qubits in the brickwork whose output she would know beforehand. Thus the computation would be private and correct provided the noise has no impact. While this is a very brief discussion of UBQC, a detailed description can be found in [4].

2.3.2 The Simulated QFHE Scheme

While UBQC is a powerful scheme, it also requires a high level of interaction between the classical and quantum machines as well as a very large number of qubits to create the brickwork state. Some of these issues are addressed with a di erent type of encryption scheme: the QFHE. Although the protocol is as of yet unpublished, this section will brie y summarise the theory behind it as developed by P. Wallden, M. Hoban, A. Gheroghiu and E. Kashe .

The idea behind QFHE is that Bob can perform an arbitrary quantum computation on data while being aware of the computation but unaware of the inputs or outputs themselves. As in UBQC, Alice prepares her single qubits and instructs Bob to entangle them. However in this case all of them are in the $|+\rangle$ state and the entanglement does not need to be a brickwork state as she does not care about revealing the nature of her computation and can save on the extra resource qubits. She also sends an extra qubit for every qubit measured at angle $_i$ 2 f $\frac{}{4}$g and instructs Bob to create a Bell Pair using them.

Bob then performs the measurements as per the MBQC pattern but does not measure each one qubit of the Bell Pairs and instead sends them back to Alice so she could measure them herself. This requires Alice to have the quantum resources to measure single qubits, but nothing beyond that.

This is because the computation lacks the interactive measurement of UBQC and instead Alice performs all of her corrections at the end of the process using the fact that Pauli corrections X and Z commute with all Cli ord operations (operations that can be simulated e ciently on a classical computer) so after each measurement the correction is teleported through to the next qubit and so all the way to the very end. This includes her classical input, meaning she can hide whether the input qubit is $|0\rangle$ or $|1\rangle$ by performing a correction at the end:

$$101!\ |i|j \quad +|j|i\ ! \quad Z_1^1 Z_2^0 Z_3^1 |+|j+|j+|i$$

This does not, however, hold true for T-gate operations, which require a measurement at angles of $\frac{}{4}$ (non-Cli ord):

$$TX^aZ^b = X^aZ^{a\ b}S^aT$$

An unwanted S-gate appears after the corrections are teleported and it needs to be removed. This is done by measuring Alice's Bell Pair qubit in either the X or Y-basis depending on the outcome of Bob's Bell Pair qubit and then using that as a correction as discussed in the next section.

Thus Bob remains unaware of the inputs and does not know all the corrections for the outputs.

2.3.3 Corrections

This section gives a quick overview of the corrections on each qubit based on the measurements before it due to the ow of the measurement pattern. s_i are measurement outcomes, b_i are corrected outcomes, calculated in increasing order. $_i$ is the measurement outcome of Alice's qubit in each Bell pair in the case that there is a qubit in the computation measured at $\frac{}{4}$. The ow is represented by $f(\ )$ and Z^i is the set of qubits that have a Z-dependence on qubit i. This means that qubits $j\ \ 2\ Z^i$ have to have a Z-correction based on the value of i. These corrections extend to the input, meaning there should be an extra layer of corrections depending on whether or not the value of the input is 1.

Bob always has to measure in default, non-corrected angles, while Alice has to measure her extra qubit in a di erent basis:

$$_i = b_{f\ \ ^1(i)}$$

where $_i = 0$ means Alice has to measure in the X-basis and $_i = 1$ means measuring in the Y-basis.

If the measurement angle in the MBQC pattern is taken to be $_i$, then the following corrections should be performed:

If $_i\ 2\ f\ 0;\ \ g$ then:

$$b_i = s_i \sum_{j\ 2\ Z^i} b_j$$

If $_i\ 2\ f\ \frac{}{2}; \frac{3}{2} g$ then:

$$b_i = s_i \quad b_{f\ \ ^1(i)} \sum_{j\ 2\ Z^i} b_j$$

If $_i\ 2\ f\ \frac{}{4} g$ then:

$$b_i = s_i \quad _i \sum_{j\ 2\ Z^i} b_j$$

3 Method

The aim of this project was to simulate a simple application of the QFHE protocol both classically and then on the IBM 16-qubit chip once beta-access was granted for its use. The classical simulations were first coded in Microsoft's LIQ$Ui|$ simulation platform for an efficient test of correctness, then adapted to the IBM Python API and its chip's hardware.

The most important aspect of the simulation was to show that a given quantum computation would produce the same probability amplitudes for each qubit with interactive corrections as with the deferred corrections and Bell pair measurements done by Alice. As discussed above, the Pauli corrections are reliant on the flow of the qubit measurement patterns[12], thus the circuits that were simulated had to have $\frac{\pi}{4}$ measurements at particular locations on the graph state in order to have an impact on the final corrected qubit amplitudes.

This means that most of the MBQC circuits derived from circuits originally composed of unitaries such as Pauli gates and T-gates would often not be as optimal as simply implementing a specially designed MBQC computation, where the position of qubits in the graph state and the angles at which they were measured could more easily be tailored to fit the parameters to be explored. The final circuit does not perform any 'useful' computation but rather works as a means to graphically and numerically verify that the protocol performs as it should.

An important consideration was the size of the circuit. Classical simulations of quantum systems require large amounts of memory and processing power in order to deal with the number of different states that are being operated on at once. This means that the circuit cannot be composed of too many qubits, else the classical simulation would not run without the help of a supercomputer. A classical computer can usually handle up to about 15-30 qubits, depending on the type of and efficiency of the implementation as well as the computer's processing power. Beyond that, however, the maximum number of qubits that could be used in this simulation would always be 16 in order to later implement it on the IBM chip. In fact, the number had to be even lower because of the adjustments and possible extra qubits required for the final implementation on the IBM machine as discussed later in the section.

After a series of trials and errors, the graph of the final circuit which was simulated can be seen in Figure 5.

Figure 5: A graph of the final MBQC circuit used in the simulation. The angles on each vertex represent the angles of measurement of each qubit and the flow is from left to right.

3.1 LIQ Ui ji

LIQ Ui ji is a simulation platform for quantum computation that operates in the F# language. Its high functionality allowed for easy and e cient implementation of di erent families of unitaries and models of quantum computation and would run simulations much faster than the IBM Python API. It was used to test many di erent candidate circuits.

Most of the tests consisted of starting out with a simple quantum circuit, then testing whether a conversion to the gate set $J(\ )$ would a ect the amplitude outpts of the simulation. The reason this was done is because the conversion from a computation composed of these unitaries to an MBQC pattern would serve as an intermediary test for correctness when switching between quantum circuits and MBQC patterns.

For each di erent type of simulation, the circuit was run 1000 times and corrected output values for each of the 3 qubits in the output layer were collected as statistics and plotted as histograms using in-built F# methods to visualise the nal probability amplitudes for each qubit after the computation. The input qubits could classically represent values 0-7 and the results were plotted for all of those inputs.

A full manual for the simulator can be found at [13] and examples of code are listed in Appendix A.

3.1.1 Interactive Corrections

The rst version of the circuit to be tested had its Pauli corrections written out explicitly as modi cations of the measurement angles based on qubits measured in the past as per the ow of the computation (see Section 2.2.2). Since LIQ Ui ji allows access to the values of single-qubit measurement outcomes as soon as the measurement is performed, it was possible to simulate the 'interactive' angle modi cations to correct outcomes of all the qubits.

Thus all that was required was to prepare all the qubits as $|+\rangle$ states by applying Hadamard gates to qubits in $|0\rangle$ states, then entangling them with $\hat{Z}$ operations as in Figure 5 and then measuring them at angles conditioned on previous measurement outcomes (by using the de nition of the modi ed angle from MBQC) or di erent classical inputs. The code for the measurement part of the circuit can be found in Appendix A.

3.1.2 Deferred Corrections and Implementing the Protocol

The next step was to remove the interactive correction method and to attempt to perform the corrections after all the qubits had been measured. The code would then only perform M_i for the that were pre-determined to perform the computation without changing them based on outcomes obtained during the run. In addition, there would be two extra client qubits added for each of the $\frac{\pi}{4}$ measurements and creating Bell Pairs between them as described in Section 2.1.1.

The client halves of the Bell Pairs would then be measured after all the server measurements were done. Finally, results for the output layer of qubits would be calculated by summing-modulo-2 all the relevant measurement outcomes for each output qubit as per the protocol. Examples of this part of the code can be found in Appendix A.

In order to check that without the corrections the server would glean no information about the computation results, values of output qubits were plotted without using any

corrections.

3.2 IBM Quantum Information Software Kit (QISKit)

This year IBM made strides to release a brand new API using the Python language to allow its clients to compose and run quantum circuits both on simulators and their 5-qubit quantum chip hardware[14]. In September, the 16-qubit chip was made available for beta-testing using their cloud server. It was now possible to simulate the QFHE scheme on a real quantum computer.

There were several issues with doing this, as the chip does not allow for networked computations, re-usability of qubits or several other functions necessary or helpful to both carry out the simulation and do it with su cient fault-tolerance. The methods to adapt the circuit used in LIQUi ji and improve its fault-tolerance through various optimization methods are described below[1].

3.2.1 Conditional Measurements

While the LIQUi ji simulator allows for a lot of freedom in extracting measurement values throughout the run of a circuit as well as an easy simulation of interactive computation, that is not the case of the IBM server. This is due to the fact that a computation has to be done on a single chip whose functionality only allows for a nal readout of the state of the system, rather than single values of qubits throughout. This creates a problem when simulating the protocol, as Alice's measurements of her Bell pair qubits are conditional on the results of the measurements that Bob has supposedly already performed and transmitted to her.

Luckily, there is a relatively simple workaround which complicates the computation only by adding a few unitaries and does not require any additional qubits. This modi - cation, in its simplest form, is a controlled phase gate $\hat{S}^y$. As discussed in section 2.4.1, we know that Alice's qubit has to be measured in a basis

$$_i = b_i \quad ^1(i)$$

With $_i = 0$ being a measurement in the X-basis and $_i = 1$ a measurement in the Y-basis. In essence her basis depends on all the corrections applied to the qubit $b_{(i)}$. In the circuit used for the simulation (Figure 5), each $_i$ is determined from a qubit measurement outcome that does not require corrections, since they are in the rst layer of qubits in the ow- the input layer. The only possible X-corrections they might require are from the classical input value of the qubit and these can be applied as X-gates once the rotation of the qubits to the required basis was performed[2].

A measurement in the Y-basis is the same as performing a rotated measurement with angle $= \frac{}{2}$. Thus we can use the qubit determining $_i$ as a control qubit that performs

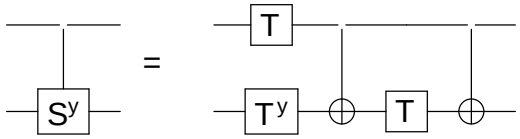
[1] More information on the QISKit software is available in their github repository: https://github.com/QISKit [Last accessed 24/11/2017]

[2] Note that if there were multiple corrections from i di erent qubits, the simple solution would be to add up their contributions by using CNOT gates with the correction-requiring qubit as the target. If its initial measurement outcome were i, each of its corrections would come from the qubits with measurement outcomes s_i. Thus the corrected measurement would be $_i = b_i \quad s_1 \quad s_2 \quad ::: \quad s_i$

a $\frac{}{2}$ rotation on Alice's qubit if its value is 1 and does nothing if it is 0. This is a gate that can be constructed using the relation[15]:

$$T^y X T X = e^{i\,=4} S$$

Normally a global phase induced on a state (such as the factor $e^{i\,=4}$ in front of the S gate) has little impact in the computation, since the observable value would remain unchanged, but since this is a controlled operation, an outcome 1 of the control qubit would give $e^{i\,=4}S$ on the target instead of the S operation that we needed. Due to this, we need to add an additional T^y gate to the control qubit, or in our case a T gate since the nal operation we are hoping to get is S^y:



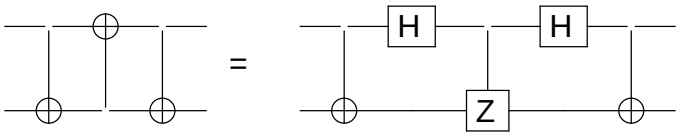
So Alice's qubit measurement is now done at the correct basis. The only thing left to take care of is to apply the corrections from the relevant qubits to the output layer, which is done using $CNOT$ gates. These are essentially controlled X- gates with the control being the qubit inducing a correction and the target being the qubit whose value needs to be ipped with the X operation. Hence the only measurements performed in this modi ed model are on the qubits in the output layer, with the others being rotated to the correct measurement basis without actually performing any measurements operations.

3.2.2 Qubit Couplings

Once the circuit is rewritten to be run on a single chip, the issue of connectivity also needs to be resolved. The IBM 16-qubit transmon chip has a speci c topology that cannot be re-arranged (see Figure 6), so any entanglements can only happen between qubits that are in the correct positions on the connectivity graph.

Figure 6: A schematic of the IBM 16-qubit chip (left) and a coupling map describing all the possible two-qubit gate connections (right), where the arrow points from the control-qubit to the target. Source: https://quantumexperience.ng.bluemix.net/qx/editor

The solution was to optimize the arrangement of all the qubits to bring the pairs requiring two-qubit gates as close together as possible and then implement SWAP-gates where required to ensure the direction of the operation and the required qubits were in the right place. The usual implementation of a SWAP gate would require a two-way control channel, but there is a modi cation that implements a SWAP using only one control-qubit:

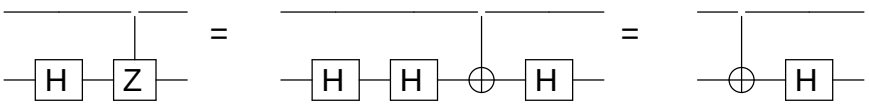


The connectivity of the chip and these methods were enough to arrange the circuit in a way that would allow for its successful execution.

3.2.3 Final Optimization and Classical Post-Quantum Processing

The above methods, while replicating the operations of the QFHE protocol, also increase the complexity of the circuit and the possibility of decoherence of the qubits as well as errors induced on them by applying gates. Thus a nal set of runs for the simulation was done with a number of optimisation methods and using classical post-processing of measurements as in the LIQ i ji case, since the Python language API that IBM has provided would allow for some access to single qubit measurements.

The optimisation of gates consisted of removing as many unnecessary SWAP operations as could be a orded (such as in the creation of Bell Pairs, where the two qubits in the pair would be identical for all intents and purposes) and removing unnecessary H operations and converting ^ Z operations into CNOTs. This was due to the fact that a ^ Z operation could e ectively be decomposed into H and CNOT gates, which is precisely how the IBM hardware would operate since its hardwired gate-set did not include ^ Z unitaries. This often led to two H gates being applied to a qubit in succession, which was e ectively just applying an Identity gate. Thus they could both be removed as long as the rest of the circuit was modi ed to not be a ected, such as in the example below:



Besides the optimisation of the circuit itself, the possibility of extracting single-qubit measurements and ridding the computation of a number of SWAP and CNOT gates which were used to apply error corrections was also implemented. The IBM QISKit API outputs its computation results as a dictionary of nal measured states of the chip and the number of times that the particular state was measured out of the number of times that the circuit was run. Through a series of trials and errors, it was possible to determine which bit in the resulting outputs corresponded to a particular measured qubit value and how they were arranged. Using this, a classical method was implemented to correct the measured outcomes as in the LIQ i ji code. This required more measurements to be performed by the hardware, but reduced the number of gates and hence the runtime by a signi cant factor. As in LIQ U i ji , all the error-inducing qubits were measured as well as the layer of output qubits. These were then summed-mod-2 in the right combinations to produce the required nal output values of the computation.

4 Results and Discussion

This section presents the numerical and graphical data was obtained from the simulations and runs on the IBM chip using the methods discussed above. It also discusses the meaning of these results in terms of the QFHE protocol as well as the hardware performance of the 16-qubit chip.

4.1 LIQ Uiji Outputs

To verify that the protocol performed as it should, simulations on LIQUiji were run a number of times while collecting di erent statistics from the points of view of Alice and Bob. This involved an unsecure computation where both were privy to the correct output (interactive corrections), as well as QFHE computations where Alice would apply the corrections to her outputs and Bob would simply get uncorrected values.

The results of some of these runs are presented below in Figures 7 and 8 as histograms produced using LIQUiji's graphing tools. Although they are each illustrative of only a single experiment, repetitions showed a standard deviation of less than 1%, so they can be considered as representative of the entire set of 20 runs that were performed.

(a) (b)

Figure 7: Graphs showing the statistical output of qubit values (y-axis) for each classical input of the computation (x-axis). The graph on the left (a) is produced using the QFHE protocol and applying corrections post-measurement where only Alice would attain the outputs shown, while the graph on the right (b) is one with interactive corrections where both Alice and Bob are aware of the inputs and outputs.

The histograms in Figure 7 are used to show that Alice gets the correct amplitudes of her qubits at the end of the computation, since her corrected values in (a) are signi cantly similar to the original unsecure computation (b). While the probabilistic nature of the computation outcomes means there is a chance she won't get the same answer on a single run, the simulation is not required to show this, since the original circuit is not meant to be a 'useful' computation. It is enough that the computations are signi cantly statistically equivalent, so as to demonstrate that if a di erent computation with a useful result were performed using the QFHE, it would still produce the same outcomes as doing it with an unsecure channel. If the results then need to be veri ed by running a circuit a number of times as was done above, then this is up to Alice.

Figure 8: Bob's statistical output (y-axis) for each classical input (x-axis) if all he has access to are the performed qubit measurement outcomes and no knowledge of the corrections or Alice's Bell Pair measurement outcomes.

Figure 8 demonstrates that without knowing the outcomes of Alice's Bell Pair qubits or the corrections to be applied due to the input layer, all Bob can really know about the computation are the outcomes of the uncorrected output qubits, which tell him nothing about what Alice actually obtained as a result.

Without the corrections, the values of output qubits are essentially random, hence they have an almost equal chance of being measured as a 0 or a 1.

4.2 IBM Simulator Outputs

The function of running the IBM simulator despite already having simulated results was to check that the translation of what was done with LIQUi|ji to the IBM platform was successful, since the two work differently in their simulation implementations and the methods provided. IBM's simulator would also predict how the circuit would be run on the real chip and so correctness was paramount.

Figure 9: A histogram showing statistical results for 1000 runs on the IBM simulation of the protocol for classical input 0. It only shows the states that were measured at least once and does not provide information on single-qubit measurements for each separate run, but rather how often a final 3-qubit state as a probability.

The classical inputs would have to be run one-by-one to reflect the performance of the real chip. Figure 10 is an example of the type of graphical representation of output data that QISKit would provide. It is clear to see, for example, that the middle qubit's value in the final measurements was always 0, since no state was ever measured where its

value was 1. But other than that, the API does not provide information on single-qubit values in separate runs of the circuit.

Single qubit measurements had to be extracted from the simulation runs by processing the numerical data for the nal output states and their frequency, which is the greatest level of access to the results of computations that the API allows for. This enabled for a better comparison to the outputs produced by LIQUi ji and were thus far more useful than the histograms that QISKit's internal methods produced. Table 1 shows the counts with post-classical processing of the output data, where an experiment with 1000 runs of the simulator for each classical input value was performed. Again, althogh the experiment was repeated 20 times, the standard deviation of the count values was less than 1% so the results for this run can be considered as representative for all the repetitions.

	Classical input value							
	0	1	2	3	4	5	6	7
Q1	128	125	134	140	890	836	854	810
Q2	0	0	1000	1000	0	0	1000	1000
Q3	130	843	140	812	147	855	127	861

Table 1: The number of times out of 1000 runs that a measured value of 1 was observed for each output qubit after the results were processed. These very clearly re ect the corrected outputs obtained in LIQ Ui ji .

A plot of the above was made using F#, in order to have a visual representation of the statistical similarities between LIQUi ji 's results and those of the IBM. It can be seen from Figures 7 and 10 that the computation is performing correctly.

Figure 10: The results shown in Table 1 represented by an F# histogram. The computation results appear to be correct when compared to the output amplitudes obtained in LIQ Ui ji .

4.3 IBM 16-Qubit Chip Outputs

Once correctness was established using the simulator, a real run could be done on the 16-qubit chip. Each separate classical input was again run 1000 times to collect data and classical post-processing extracted the values for the relevant output qubits which are plotted in Figure 11.

Figure 11: Results of all initial classical input runs on the 16-qubit chip as plotted with F#. The e ects of decoherence are apparent.

It is clear that the e ects of decoherence are signi cant. The statistical pattern that was apparent in the simulation is almost non-existent here, with the qubit amplitude probabilities generally close to 0.5 on average. This is made more apparent when looking at the numerical values of the output qubits as displayed in Table 2.

	Classical input value							
	0	1	2	3	4	5	6	7
Q1	478	477	502	488	447	466	445	477
Q2	333	378	613	591	399	338	516	557
Q3	448	482	470	460	458	484	478	469

Table 2: The number of times out of 1000 runs that a value of 1 was measured on the 16-qubit chip. Qubits 1 and 3 appear to have completely decohered, with their amplitudes being around 0.5 on average. The middle qubit still appears to show some variation but it is not nearly signi cant enough to be treated as a useful result.

The average deviation from the values in the simulation was calculated to be around 37.2%. This is disappointing and yet not unexpected, as IBM provides data on the average errors of all operations for each qubit on the chip.[3] These seem to imply that the magnitude of the error is to be expected for a circuit with over 100 unitary operations, many of which are applied between 2 qubits. The average error for a single gate is predicted to be of the order of 10^3 while multi-qubit gates and read-outs will cause errors of magnitude 10^2. This, together with the time it takes for the computation to be completed (during which the qubits continue to decohere until they are measured), is likely to cause enough of an error to just reduce each qubit's measured value to random chance.

[3] https://quantumexperience.ng.bluemix.net/qx/editor [Last accessed 26/11/2017]

Figure 12: An example of the histograms produced using QISKit's internal methods. While it does not provide much information regarding the single-qubit outputs, it allows for a visualisation of the lack of statistical variation between di erent output states (as given by the x-axis).

Unfortunately, due to restrictions IBM places on the number of times an experiment can be run on its chip per day, each run could only be repeated two to three times, but again the standard deviation from the mean was less than 1%, making it very likely that the magnitude of the errors would persist regardless of repetition.

4.4 Results Post-Optimization and Using Classical Processing of Measurements

The results obtained as described above prompted a series of attempts at optimization as described in theMethods section. The results appear promising, as seen in Figure 13, where despite the lack of improvement in qubits 1 and 3, the deviation of qubit 2 from the simulation values was reduced from 39.6% to 25.2%.

Figure 13: A histogram of optimized IBM chip output qubit values for each classical input. The experiment was again run 1000 times for each input and the extracted values plotted using F#.

This can easily be attributed to the fact that the second qubit has a far smaller number of operations when calculating its corrected output value. It does not require to wait for Alice's qubit measurements or the extra gates that are needed to perform these (refer back to the original MBQC circuit diagram in Figure 5).

	Classical input value							
	0	1	2	3	4	5	6	7
Q1	492	512	532	512	557	490	548	501
Q2	252	265	759	738	261	268	762	772
Q3	471	492	490	497	480	510	503	500

Table 3: Table showing the number of times out of 1000 that a 1 was calculated as the value of output qubits in the optimized version of the IBM circuit. While the second qubit shows some statistical improvement, the computation as a whole does not provide useful results.

It is di cult to determine how much of the nal error was due to decoherence over time and how much of it was due to the number of operations applied to each qubit. It is likely that the nal circuit runs, as optimized internally by QISKit, were the best possible version of results that could be produced. An attempt to add barriers, which act by con ning operations to a particular order of execution only made the results for the second qubit worse, meaning the small advantage of performing some operations on qubits 1 and 3 sooner came at the cost of the coherence of the 2nd with no tangible gain for the coherence of the other two. Further attempts at optimization by moving the measurements of the client to be before the other measurements and similar methods all seem to produce the same results, so nothing else could be done to optimize the results.

5 Conclusions and Future Research

The data obtained during the experimental runs on the IBM chip and the classical simulations lends itself to several di erent conclusions, both concerning the QFHE protocol and the future of quantum computation as a commercial service. One positive result is the success of classical simulations. While this does not re ect the di culty of implementing it in real quantum hardware architectures, it does at least show that the scheme can perform as intended, in theory, on simple computations. There is little reason to doubt that the transfer of this functionality to real technology is possible, if the technology is to perform as predicted. The problems concerning this, however, merit some discussion.

IBM's current quantum technology does not seem to provide enough fault-tolerance to test the QFHE protocol. The issue of decoherence and the errors induced in computations through performing operations such as gates and measurements will likely be one of the greatest challenges in the future of quantum computation. The simulations through classical means, both using LIQ i ji and the QISKit simulator exist to show that, within the con nes of quantum technology performing as it should with 100% fault-tolerance, the protocol should work as intended. The fact that the real chip does not con rm this can only be explained as either quantum operations in the hardware not being equivalent to those in theory, as used by the simulators, or through the fragility of quantum systems and the high level of errors induced in these computations without the resources to

error-correct. The former, while not impossible, would still imply that something about these technologies is inherently di erent to what the theory predicts and makes less sense when considering that smaller examples of code using the same technology and the same theoretical premises appear to produce correct results as in example circuits provided by the QISKit software package[4]. However this is part of a larger problem of quantum veri cation which is too vast to be discussed here (a recent review paper on the subject gives a good overview of the current state of quantum veri cation research: [15]). If increasing the scale of the computation begins to make it fail, it's far more likely to be a growth in error than the theoretical premises failing, partcularly for examples of this size. Presuming this allows for a discussion about the cause of these errors and how to better simulate and nally apply the QFHE protocol once the technological resources become available.

The obvious solution, beyond improving the fault-tolerance of the quantum hardware itself, is to implement error-correcting codes. Shor's code in particular is promising- it can correct any corrupting channel as long as only one qubit is a ected. However, it also requires eight additional qubits for one qubit of useful information[8], so IBM's current 16-qubit chip does not have nearly enough in terms of resources to involve this type of correction and perform a useful computation, although once it does, the same example circuit could be run to check whether there is any development.

The second possibility of improvement is that of avoiding decoherence by extracting information from qubits as quickly as possible. While remaining errors are unavoidable once the number of operations in the circuit has been optimized, giving it less time to lose information through contact with the environment is a promising option. The most likely reason that the rst and third output qubits give the worst results is because of the time it takes between the initiation of the Bell Pairs and their actual measurement- as well as the number of gates the conditional measurement requires. In the case of QFHE, the client would supposedly be able to measure their qubit immediately after the server has measured their half and sent it back, reducing the amount of time for decoherence (if the transmission of the qubit does not make the loss of information even worse than in the current experiment).

Furthermore, the general idea of QFHE as well as other MBQC-based protocols is that the qubits could be entangled and measured layer-by-layer along the direction of the ow. This means that the time between initializing and measuring the qubit could be made as short as the span of two entanglements and 2-3 gate operations. This is, in fact, possible with the current hardware, but since the computation is not actually networked, Alice and Bob cannot be performing measurement simultaneously, leaving the measurement of the Bell pair qubit to accumulate decoherence-induced errors.

An interesting possibility to consider, once the hardware for it has been developed, is the reusal of qubits in MBQC computations after they'd been measured. If a true networked computation were a possibility, then depending on the dimensions of the computation, each layer of the MBQC graph could be re-initialised once the qubits in it have been measured and the values stored or transmitted. This would allow for in nite depth in a computation with no marginal increase in decoherence of each qubit.

Finally, although this has already been alluded to above, it is important to discuss

[4] https://github.com/QISKit/qiskit-sdk-py/tree/master/examples/python [Last accessed: 26/11/2017]

how the experiments run on the IBM chip do not re ect the actual implementation of QFHE in a number of important ways. The classical simulations are a better re ection bar the errors and decoherence, but the actual hardware results are for a single circuit that has been transformed in so many ways that the original scheme is barely recognizable. The fact that the client cannot measure their Bell pair half is important not just in terms of reducing the decoherence time, but also in predicting what kind of impact the transmission of the qubit to her will have. If the errors in gate operations and measurements are this large, should future quantum hardware be adapted to reduce the errors of qubit transmission and reception as well? There is promising technology already being developed, such as by researchers in DELFT, who have managed to entangle two electrons separated by a distance of 3 meters using spin-photons[16]. Perhaps these sort of quantum networks will be a far better place to test and implement security protocols based in MBQC.

The development of quantum information hardware is currently moving at an un-precedented pace, with IBM set to release a 20-qubit chip by the end of the year and a 50-qubit device soon after[17]. Despite the shortcomings apparent in current technologies, it looks to be a bright future for companies like IBM and Google that continue to invest in making their devices more fault-tolerant and adding functionality. Once the quantum resources exist, there is every reason to expect that protocols like UBQC and QFHE will be used extensively in areas ranging from quantum encryption to the development and understanding of new pharmaceuticals and simulations of molecules.

6 Acknowledgements

I would like to thank Elham Kashe for giving me the chance to investigate the most fascinating, cutting-edge technologies ever developed as well as Petros Wallden for all the help in understanding his protocol as well as being a great professor. Most of all, thank you to Andru Gheorghiu without whom I would have barely grazed the surface both in understanding the concepts for the project and in actually being able to execute it. Like a padawan following the jedi master, I can only hope to continue the legacy.

References

[1] S. Lloyd, \Universal quantum simulators", Science 273, 1073{1078 (1996).

[2] P. W. Shor, \Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM Journal on Computing 26, 1484{1509 (1997).

[3] C. Vu, Ibm builds its most powerful universal quantum computing processor, [Online; posted 17-May-2017], http://www-03.ibm.com/press/us/en/pressrelease/52403.wss, May 2017.

[4] A. Broadbent, J. Fitzsimons, and E. Kashe , \Universal blind quantum computation", ArXiv e-prints (2008).

[5] V. Dunjko, J. F. Fitzsimons, C. Portmann, and R. Renner, \Composable security of delegated quantum computation", ArXiv e-prints (2013).

[6] D. Wecker and K. M. Svore, \Liqui ji : a software design architecture and domain-speci c language for quantum computing", Feb. 2014.

[7] T. Dass, \Measurements and Decoherence", eprint arXiv:quant-ph/0505070 (2005).

[8] M. Nielsen and I. Chuang, Quantum computation and quantum information, Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2000).

[9] R. Perry, Quantum computing from the ground up (World Scienti c, 2012).

[10] P. Wallden, Introduction to quantum computing, University Lectures, https://qcintro.wordpress.com/lectures/ , 2017.

[11] V. Danos, E. Kashe , and P. Panangaden, \Parsimonious and robust realizations of unitary maps in the one-way model", 72, 064301, 064301 (2005).

[12] V. Danos and E. Kashe , \Determinism in the one-way model", 74, 052310, 052310 (2006).

[13] D. Wecker, Liqui ji users manual, [Online], http://stationq.github.io/Liquid/docs/LIQUiD.pdf , 2015/16.

[14] L. S. Bishop, Developers wanted: quantum computing, python, and jupyter notebooks, [Online; posted 10-March-2017], https://developer.ibm.com/code/2017/03/10/developers-wanted-quantum-computing-python-jupyter-notebooks/ , Mar. 2017.

[15] A. Gheorghiu, T. Kapourniotis, and E. Kashe , \Veri cation of quantum computation: an overview of existing approaches", (2017).

[16] H. Bernien, B. Hensen, W. Pfa , G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson, \Heralded entanglement between solid-state qubits separated by three metres", Nature 497, 86 EP - (2013).

[17] C. Vu, Ibm announces advances to ibm quantum systems ecosystem, [Online; posted 10-November-2017], http://www-03.ibm.com/press/us/en/pressrelease/53374.wss, Nov. 2017.

Appendices

A Samples of Code

A.1 QISKit Code for IBM

Listing 1: Example of extracting relevant data from the IBM chip output and processing it to get corrected single-qubit outputs in the nal, optimized version of the circuit. The states and their frequency are initially given as string-integer pairs in a Dictionary object.

```
# RUN THE EXPERIMENT
result = qp.execute(['MBQC'], backend='ibmqx5', shots = 1000, timeout = 600)

# get the data obtained in the experiment
counts = result.get_data('MBQC')
#ran_qasm = result.get_ran_qasm('MBQC')
#print(ran_qasm)

# Extract relevant counts for qubit outcomes from the data obtained
counts = counts['counts']
counts = counts.items()
for i in counts:
    val = i[0] #state measured
    times = i[1] #times it was measured
    if (times != 0):
        out0 = ( ord (val[0]) 48)
        out2 = ( ord (val[2]) 48)
        out3 = ( ord (val[3]) 48)
        out5 = ( ord (val[5]) 48)
        out6 = ( ord (val[6]) 48)
        out7 = ( ord (val[7]) 48)
        out8 = ( ord (val[8]) 48)
        out9 = ( ord (val[9]) 48)
        out10 = ( ord (val[10]) 48)
        out12 = ( ord (val[12]) 48)
        out13 = ( ord (val[13]) 48)
        out15 = ( ord (val[15]) 48)

        # get corrected final outcomes
        res1 += ((out15 + out0 + out12 + out13)%2)    times
        res2 += ((out8 + out7 + out6)%2)    times
        res3 += ((out2 + out3 + out10 + out9)%2)    times
```

Listing 2: Sample of optimized code for the IBM chip, wherein many Hadamard operations have been removed and subsequently entanglements have been de-constructed into CNOT and Hadamard operations.

```
# Hadamard everything
mbqc.h(q1[0])
mbqc.h(q1[1])
mbqc.h(q1[3])
mbqc.h(q1[4])
mbqc.h(q1[6])
mbqc.h(q1[8])
mbqc.h(q1[9])
mbqc.h(q1[12])
mbqc.h(q1[15])

# first Bell pair
mbqc.cx(q1[15], q1[2])
mbqc.x(q1[2])
#mbqc.barrier()

# second Bell pair
mbqc.cx(q1[12], q1[5])
mbqc.x(q1[5])
#mbqc.barrier()
```

```
# entangle qubits

mbqc.cz(q1[15], q1[0])
mbqc.cx(q1[8], q1[7])
mbqc.h(q1[7])
mbqc.cx(q1[12], q1[13])
mbqc.h(q1[13])

mbqc.cx(q1[15], q1[14])
mbqc.cx(q1[7], q1[10])
mbqc.cx(q1[12], q1[11])

mbqc.cx(q1[3], q1[14])
mbqc.cx(q1[9], q1[10])
mbqc.cx(q1[6], q1[11])

mbqc.h(q1[11])
mbqc.cx(q1[11], q1[10])
mbqc.h(q1[11])
```

A.2 LIQ$Ui|\rangle$ Code

```
// adaptive measurements which include corrections
rotatedMeasurement qs.[0] (Math.PI/2.)
rotatedMeasurement qs.[1] (Math.PI/2.)
rotatedMeasurement qs.[2] (Math.PI/2.)

rotatedMeasurement qs.[3] (((-1.) ** (float qs.[0].Bit.v)) * Math.PI/4.)
rotatedMeasurement qs.[4] (((-1.) ** (float qs.[1].Bit.v)) * Math.PI/2.)
rotatedMeasurement qs.[5] (((-1.) ** (float qs.[2].Bit.v)) * Math.PI/4.)

rotatedMeasurement qs.[6] (((-1.) ** (float qs.[3].Bit.v)) * 0. + ((float qs.[0].Bit.v) * Math.PI))
rotatedMeasurement qs.[7] (((-1.) ** (float qs.[4].Bit.v)) * 0. + ((float qs.[1].Bit.v) * Math.PI))
rotatedMeasurement qs.[8] (((-1.) ** (float qs.[5].Bit.v)) * 0. + ((float qs.[2].Bit.v) * Math.PI))

rotatedMeasurement qs.[9] (((-1.) ** (float qs.[6].Bit.v)) * 0. + ((float qs.[3].Bit.v) * Math.PI))
rotatedMeasurement qs.[10] (((-1.) ** (float qs.[7].Bit.v)) * 0. + ((float qs.[4].Bit.v) * Math.PI))
rotatedMeasurement qs.[11] (((-1.) ** (float qs.[8].Bit.v)) * 0. + ((float qs.[5].Bit.v) * Math.PI))
```

Figure 14: Code for performing corrections interactively with consequent angles based on measurement outcomes of previous qubits. The method `rotatedMeasurement` was written speci cally for the simulation and takes two arguments: the qubit to operate on and the angle at which the measurement operation should be performed.

```
// Measurements with no corrections
rotatedMeasurement qs.[0] (Math.PI/2.)
rotatedMeasurement qs.[1] (Math.PI/2.)
rotatedMeasurement qs.[2] (Math.PI/2.)

rotatedMeasurement qs.[3] (Math.PI/4.)
rotatedMeasurement qs.[4] (Math.PI/2.)
rotatedMeasurement qs.[5] (Math.PI/4.)

rotatedMeasurement qs.[6] (0.)
rotatedMeasurement qs.[7] (0.)
rotatedMeasurement qs.[8] (0.)

rotatedMeasurement qs.[9] (0.)
rotatedMeasurement qs.[10] (0.)
rotatedMeasurement qs.[11] (0.)
```

Figure 15: Code of the measurements of the same qubits at the default angles provided at the beginning of the computation without any interactive corrections.

```
            // Client Measurements
            if (pads.[0] + qs.[0].Bit.v) % 2 = 0 then
                rotatedMeasurement qs.[12] 0.
            else
                rotatedMeasurement qs.[12] (Math.PI/2.)

            if (pads.[2] + qs.[2].Bit.v) % 2 = 0 then
                rotatedMeasurement qs.[13] 0.
            else
                rotatedMeasurement qs.[13] (Math.PI/2.)

            // Measurement corrections as outcomes of the relevant qubits
            let a = (pads.[0] + qs.[3].Bit.v + qs.[9].Bit.v + qs.[12].Bit.v) % 2
            let b = (qs.[1].Bit.v + qs.[4].Bit.v + qs.[10].Bit.v) % 2
            let c = (pads.[2] + qs.[5].Bit.v + qs.[11].Bit.v + qs.[13].Bit.v) % 2

```

Figure 16: Measurements of the Bell pair qubits performed by the client and the subsquent nal amplitudes of the relevant qubits calculated with deferred corrections. Variables a, b and c are then the nal corrected outcomes of output qubits 1, 2 and 3 respectively.